NASA-CR-196262

University of Southern California

IN TERIM
IN-61-CR
OCIT.
16381
7P

(NASA-CR-196262) DISTRIBUTED VIRTUAL SYSTEM (DIVIRS) PROJECT Semiannual Progress Report No. 12, 1 Nov. 1993 - 30 Apr. 1994 (University of Southern California) 7 p

N94-37509

Unclas

G3/61 0016381

INFORMATION SCIENCES INSTITUTE ISI

310/822-1511

4676 Admiralty Way/Marina del Rey/California 90292-6695

University of Southern California
Department of Contracts and Grants
Los Angeles, CA 90089-1147

DIstributed VIRtual System (DIVIRS) Project

formerly

Center for Experimental Research in Parallel Algorithms, Software, and Systems

Semiannual Progress Report #12
June 1994

Principal Investigator:
Herbert Schorr
Co-principal Investigator
B. Clifford Neuman
USC/Information Sciences Institute

Prepared under NASA Cooperative Agreement NCC 2-539
for Henry Lum, Technical Officer
NASA Information Sciences Division 244-7

Semiannual Progress Report

Covers period 1 November 1993 through 30 April 1994

As outlined in our continuation proposal 92-ISI-50R (revised) on NASA cooperative agreement NCC 2-539, we are (1) developing software, including a system manager and a job manager, that will manage available resources and that will enable programmers to develop and execute parallel applications in terms of a virtual configuration of processors, hiding the mapping to physical nodes; (2) developing communications routines that support the abstractions implemented in item one; (3) continuing the development of file and information systems based on the Virtual System Model; and (4) incorporating appropriate security measures to allow the mechanisms developed in items 1 through 3 to be used on an open network.

The goal throughout our work is to provide a uniform model that can be applied to both parallel and distributed systems. We believe that multiprocessor systems should exist in the context of distributed systems, allowing them to be more easily shared by those that need them. Our work provides the mechanisms through which nodes on multiprocessors are allocated to jobs running within the distributed system and the mechanisms through which files needed by those jobs can be located and accessed.

The Prospero Resource Manager

Conventional techniques for managing resources in parallel systems perform poorly in large distributed systems. To manage resources in distributed parallel systems, we have developed resource management tools that manage resources at two levels: allocating system resources to jobs as needed (a job is a collection of tasks working together), and separately managing the resources assigned to each job. The Prospero Resource Manager (PRM) presents a uniform and scalable model for scheduling tasks in parallel and distributed systems. PRM provides the mechanisms through which nodes on multiprocessors can be allocated to jobs running within an extremely large distributed system.

The common approach of using a single resource manager to manage all resources in a large system is not practical. As the system grows, a single resource manager becomes a bottleneck. Even within large local multiprocessor systems the number of resources to be managed can adversely affect performance. As a distributed system scales geographically and administratively, additional problems arise, such as latency, trust and security.

PRM addresses the bottleneck problems by using multiple resource managers, each controlling a subset of the resources in the system, independent of other managers of the same type. The functions of resource management are distributed across three types of managers: system managers, job managers, and node managers. The complexity of these management roles is reduced because each is designed to utilize information at an appropriate level of abstraction.

The current implementation of the Prospero Resource Manager runs on a collection of Sun-3, Sun SPARC, and HP9000/700 workstations running various versions of the Unix operating system, and a single Intel486 personal computer running Mach. Communication between the job, system, and node managers, and between tasks in a job is supported by a reliable delivery protocol based on the user datagram protocol (UDP) running over local and wide-area networks. Heterogeneous execution environments are supported - a system manager may manage nodes of more than on processor type. In the common case, there is one system manager for each site. For example, our setup consists of one system manager responsible for a set of SPARCstations on USC's main campus, another managing a collection of Sun-3, SPARC, and HP700 workstations at ISI, 15 miles away from the main campus, while a third manages a set of HP700 workstations across the country at MIT. We have run applications that use processors at all three sites.

Programmers link executables for their tasks with the communication library we provide. Depending on how PRM has been configured, users then create a job description file or they make suitable entries in the Prospero Directory Service. To run a parallel application, users invoke the job manager passing it the name of the application. I/O to the terminal and to files that are not otherwise accessible to the application is handled through an I/O task that runs on the user's workstation.

During the reporting period, ISI added support for suspension and subsequent migration of tasks running under PRM. To suspend a task, the execution state is captured in a checkpoint file and the task is killed. To migrate the task, checkpoint is created by the node manager, which informs the concerned job manager. The job manager then acquires another processor from the system manager and requests the new node manager to restart the task from the checkpoint. Our implementation builds upon the checkpointing mechanisms from the Condor package of the University of Wisconsin. We have integrated this package with PRM and enhanced PRM's communication libraries to correctly handle messages addressed to migrating/migrated tasks. We have also implemented an alternative task suspension/resumption method (but not migration) for programs that cannot be relinked with Condor's checkpointing library.

We also extended the Prospero Resource Manager to support remote execution of programs that are not written specifically for PRM or relinked with the PRM library. Sequential applications, including those for which source code is not available, can now be run under PRM. Interactions between the user and unmodified sequential jobs are supported by redirecting stdin, stdout and stderr streams through a separate task running on the same. This task accepts output from the unmodified task and sends it to the terminal-I/O task and it accept input from the terminal I/O task and writes to the stdin stream of the unmodified task. A parallel application may be composed of both unmodified and relinked program modules. For such applications, a user-designated task receives terminal input; unmodified tasks communicate through their standard I/O streams as described above, and relinked tasks use PRM's message passing mechanisms.

In support of playback debugging, we are creating an instrumented version of the communication library. Instrumented send and receive functions log the communication activity of each task, enabling a debugger to examine the significant events in a task's history and exactly recreate the sequence of events.

A paper describing the Prospero Resource Manager was written and accepted for publication in the journal *Concurrency: Practice and Experience*. A copy of that paper will be included in the next report, after the issue has gone to press.

Our future plans include continued development or PRM, simplifying the installation and configuration procedures, adding support for additional hardware platforms, and deploying PRM to additional sites on the Internet. Work continues on debugging tools. We are also moving some of the security services used by the Prospero Directory Service into the reliable delivery protocol on which PRM is also based. This will provide stronger integration with the security services used in other parts of the project.

The Prospero File System and Directory Service

During the reporting period, we continued development of the Prospero File System and Directory Service, a file system and directory service based on the Virtual System Model. Most of our development was directed toward moving Prospero from a prototype to a production system. This included the addition of support for multi-threading of the Prospero server, and caching of directories and attributes.

On the technology transfer front, negotiations were completed with the Open Computing Security Group (OCSG) who has licenced Prospero from USC and intends to integrate it with several of their products. Also, America Online (AOL) introduced their Internet service, allowing AOL users to access information from Gopher, WAIS, and Prospero servers on the Internet. AOL's information gateway, developed by Pandora systems, uses a Prospero server that translates data from the Gopher and WAIS formats, returning data to a Prospero client using the Prospero protocol. The Prospero server caches data from other services, reducing network and server load, and improving performance and reliability. The Prospero gateway handles roughly 100,000 queries per day, and has a cache hit ratio of 88 percent. Our work to optimize performance of the Prospero server was driven, in part, by the requirements for production use by AOL, but the improvements were incorporated into the standard Prospero release and, therefore, benefit all users of Prospero.

We continued development of a transitive indexing system. Transitive indexing is a scalable technique for generating high-level indices that direct users to information of interest. An indexer was completed that when executed with a depth and a list of attributes to be indexed as arguments, builds an index by traversing the Prospero directory structure and retrieving the values of the named attribute from each file, directory, or object. An index is then generated mapping values of the attribute to the first level links through which objects with matching attribute values may be found. A reference to this index is bound as an attribute to the node from with it was generated. When generating an index, if a node is reached that already has an associated transitive index for the same attribute and at least the requested depth, the data from the existing index is propagated upward and the traversal is pruned at that point.

Information from a transitive index is obtained using the TQ filter on a Prospero directory query. When querying a directory, Prospero allows the user to specify filters that modify the list of objects that are returned. By specifying an attribute and a value as arguments to the TQ filter, the list of links that appear in a directory will be restricted to only those subdirectories through which objects with the requested attributes can be reached.

Our plans for future development include continued work on performance and caching, integration of Prospero and Mosaic, more work on transitive indexing, and improved release engineering.

Security for Distributed Systems

The widespread use of the computing and information infrastructure we are developing as part of the DIVIRS project requires an underlying security infrastructure to provide fine-grained access control mechanisms to protect computing and information resources and accounting mechanisms to manage their use. Work on such a security infrastructure was selected for funding under another project at ISI. As part of the DIVIRS project we have implemented the Prospero Resource Manager and the Prospero Directory Service so that it can take advantage of such infrastructure when it becomes available.

Electronic Commerce

To support for-hire services, it must be possible to send payments over the network when information or other on-line services are requested. As part of an AASERT award attached to the DIVIRS contract, which provides funding for an additional graduate student, Gennady Medvinsky has been working on electronic payment mechanisms for the Internet. In a paper presented at the ACM Conference on Computer and Communications Security, we describe how electronic currency can also be implemented on top of the security infrastructure just discussed. He has begun implementation of the mechanisms described in the paper.

Publications

The following papers were prepared, or presented during the reporting period.

Gennady Medvinsky and B. Clifford Neuman. NetCash: A design for practical electronic currency on the Internet. In *Proceedings of the first ACM Conference on Computer and Communications Security.* Fairfax VA, November 1993. (copy included with last semi-annual report)

B. Clifford Neuman and Santosh Rao. The Prospero Resource Manager: A scalable framework for processor allocation in distributed systems. To Appear *Concurrency: Practice and Experience.* Summer 1994. (copy to be included with next semi-annual report)

APPENDIX A - GLOSSARY

ACM	Association for Computing Machinery
AOL	America Online
ARPA	Advanced Research Projects Agency
DIVIRS	Distributed Virtual Systems
INET'93	The 1993 annual conference of the Internet Society
I/O	Input/Output
ISI	Information Sciences Institute
MIT	Massachusetts Institute of Technology
OCSG	Open Computing Security Group
PRM	Prospero Resource Manager
PVM	Parallel Virtual Machine
TQ	Transitive query
UDP	User Datagram Protocol
USC	University of Southern California

INFORMATION
SCIENCES
INSTITUTE

4676 Admiralty Way/Marina del Rey/California 90292-6695